

DISCOVERING INTERNAL REPRESENTATIONS FROM OBJECT-CNNs USING POPULATION ENCODING

Jianyu Wang¹, Zhishuai Zhang^{2,3}, Vittal Premachandran¹, Alan Yuille¹

¹University of California Los Angeles,

²University of Science and Technology of China, ³Yitu Technology

{wjyouch@, vittalp@, yuille@stat.}ucla.edu, {baird}@mail.ustc.edu.cn

ABSTRACT

In this paper, we provide an approach for understanding the internal representations of Convolutional Neural Networks (CNNs) for detecting objects. We hypothesize that the CNNs represent object parts at different levels of the network by populations of neurons and propose a simple clustering technique to extract these representations, which we call *visual concepts*. We visualize these visual concepts and show that they are typically very tight (in the sense that corresponding image patches appear visually very similar) and give a taxonomy showing that they represent most object parts. We show that visual concepts yield much tighter patches than those which are associated to individual filters (i.e. those patches which most strongly activated the filter). For more quantitative experiments, we treat visual concepts as unsupervised part detectors and evaluate them for the task of key point detection on the PASCAL3D+ dataset. These studies quantitatively show the advantages of population encoding, compared to single-filter detectors, for task of object-part detection. Moreover, these methods are only slightly worse than fully supervised methods which train classifiers using knowledge of the keypoint locations. We also perform some preliminary experiments where we uncover the compositional relations between the adjacent layers using the parts detected by population encoding clusters.

1 INTRODUCTION

Deep Neural Networks have been greatly successful at performing a range of visual tasks such as image classification, as showcased by Krizhevsky et al. (2012) and Simonyan & Zisserman (2015), at object detection as shown by Girshick et al. (2014), and, at semantic segmentation as demonstrated by Long et al. (2015) and Chen et al. (2015). But despite these successes there is limited understanding of why CNNs work so well. In particular, it would be nice to “read the entrails” of the deep networks and understand what the hidden units are doing. A related question, which we address in this paper, is to determine whether CNNs build internal representations of object parts and, if so, how these representations are encoded by the neurons in the network. Understanding these issues will lead to better CNN architectures and can also be used as an unsupervised way for learning object parts (by unsupervised, we mean that the object class is known, but the locations and number of object parts is not).

Much work on understanding CNNs has concentrated on visualizing the activities of neurons *independently*, as done by Zeiler & Fergus (2014), Yosinski et al. (2015) and Zhou et al. (2015). This is partly inspired by conjectures about how real neurons encode concepts. In particular, Barlow (1972) suggested that each concept might be represented by a single neuron, the so-called “grandmother cell”. Our work, however, follows an alternative neuroscience hypothesis that concepts are encoded by a population of neurons (Georgopoulos et al. (1986)). This is plausible since complexity arguments suggest that neural populations are more effective than single neurons for representing large numbers of object parts. There is already some work which studies CNNs from this population code perspective as we describe in the related work in Section 2.

In this work, we explore the idea that CNNs represent parts of objects by populations of neurons. This is a fairly natural conjecture for CNNs since they contain filterbanks of many neurons (e.g.,

256 or 512) which act in the same image regions¹. To extract these representations we use clustering methods to extract visual concepts. More specifically, we use a CNN pre-trained on ImageNet and apply it to images from a single object class, such as cars or airplanes. At each level of the CNN we cluster the feature vector responses into a set of *visual concepts* given by the means of the clusters. These visual clusters typically contain between 6 and 10 filters which respond strongly. We give the details of our approach in Section 3.

We first study visual concepts by visualization. We show that most of the visual concepts are tight in the sense that they correspond to image patches which are visually very similar and often correspond to semantically meaningful parts of the object (e.g. car tires, aeroplane wing tips, license plates, etc.). We give a taxonomy of visual concepts showing that they are complete in the sense that they cover most of the object. We compare this to the image patches which correspond to individual filters (those which cause the filter to fire most strongly) and show that although some filters are tight (e.g., for car tires) many are not. We also study how visual concepts at different levels of the hierarchy represent different parts of the objects. This is shown in Section 4. We have created a project webpage (<http://ccvl.stat.ucla.edu/projects/visual-concepts/>) which exhaustively shows visual concepts and filters for cars at pool4, and gives a taxonomy of the visual concepts.

Next we quantify visual concepts by considering them in terms of unsupervised part detection. Using the key point annotations from PASCAL3D+ dataset, we evaluate each cluster as key point detector, and select the best cluster for each key point. We adopt the precision/recall evaluation and calculate the average precision (AP) as is commonly used in object detection literature. We show that the visual concepts give fairly good average precision for part detection and outperform methods based on single filters. We compare both approaches to a supervised method which knows the location of the key points and trains classifiers to detect them using the deep network features. This supervised method, which is similar to (Long et al. (2014)), usually outperform our unsupervised approach but not by large margins (and occasionally, we do better). Finally, we explore the compositional relations between two neighboring layers of CNN. This is described in Section 5 and Section 6.

We organize the rest of the paper as follows. In Section 2, we discuss related work that aims to understand the workings of CNNs. The details of population encoding are presented in Section 3. We compare the population encoding and single filter encoding by visualization in Section 4. We describe a quantitative evaluation strategy using key points in Section 5 and present the results in Section 6. In Section 7, we present some preliminary experiments that showcase the compositionality of higher level parts using the sub-parts learnt at lower levels. Finally, we conclude the paper in Section 8 by pointing to future research directions. In the supplementary material we give results on PASCAL VOC and the effects of normalization as a preprocessing step.

2 RELATED WORK

Several works have attempted to understand deep networks through visualization. Erhan et al. (2009) used gradient descent in image space to find images that maximize the neuron activities. Zeiler & Fergus (2014) proposed a DeConvNet-based visualization strategy that mapped intermediate features activations back to the image space using the Deconvolution Networks of Zeiler et al. (2011). Similar to Erhan et al. (2009), Simonyan et al. (2014) proposed a gradient-based visualization technique to generate images by maximizing an object class score; this enabled them to visualize the notion of an object class as captured by a CNN. Building upon the idea of gradient-based image generation, Yosinski et al. (2015) incorporated stronger priors on the optimization procedure to generate “better-looking” images. They also release a toolbox that helps visualize what any individual neuron at any intermediate layer likes to see. Mahendran & Vedaldi (2015) showed that incorporating strong natural image priors to the optimization procedure produces images that look more like real images. Mahendran & Vedaldi (2015)’s optimization procedure differs from others in that they do not aim to produce an image that maximizes the activation of an individual neuron, but aim to produce natural-looking images whose bank of filter responses in a particular layer are similar to the bank of filter responses from a target image. Although they use a population of filter responses, their aim is in producing better visualizations. By contrast, we propose a method to extract knowledge,

¹We use the terms neurons and filters interchangeably in this paper.

distributed over a population of neurons, which can be used for other purposes such as unsupervised part detection.

Other works probe CNNs using by other approaches. Szegedy et al. (2013) showed the brittleness of CNNs by showing that one can change the CNN’s prediction by adding an imperceptible amount of adversarial noise to the input image. They also suggested that it might be the space of the features, and not the individual neurons, that contains the semantic information at the higher layers of the network. In related work, Nguyen et al. (2015) generated images using evolutionary algorithms that are unrecognizable to humans but can easily fool CNNs.

Although the concept of distributed representations has been popular in the neuroscience community, there have been comparatively few studies of it for CNNs. One work that made a preliminary attempt at quantifying the distributedness of information in CNNs is that of Agrawal et al. (2014). They question the presence of grandmother cells by computing the average precision when using individual filters as object detectors. They claim that grandmother cell-like filters may exist only for a few object classes. In order to test whether information is distributed, they trained linear SVMs on features extracted from a bank of filters after spatial pooling. Their results show that several filters are needed for achieving good performance on a number of classes, hence arguing in favor of a distributed code. But our work, however, differs from theirs in many ways; i) we extract the distributed representation using population encoding in a completely unsupervised fashion while Agrawal et al. (2014) employ a supervised technique and train SVMs on banks of filter responses after spatial pooling, ii) while Agrawal et al. (2014) restrict themselves to extracting object-level representations, we go over multiple layers of the CNN and identify the mid-level object parts/sub-parts, which is more challenging, and, iii) our visualization and evaluation methods are very different.

Discovering mid-level patches has, of course, been studied before CNNs (e.g. Singh et al. (2012) and Juneja et al. (2013)). More recently, Li et al. (2015) used CNN features with association rule algorithm to discover mid-level visual elements, and evaluated them on object/scene classification task. Their method uses a CNN only as a feature extractor without exploring how the intermediate layer CNN filters capture the mid-level visual elements. Xiao et al. (2015) and Simon & Rodner (2015) used the activations from the intermediate layers of the CNN to find object parts, which they then use in the task of fine-grained object classification. This method was based on the assumption that a single filter from an intermediate layer can detect parts. Zhou et al. (2015), looked at the hierarchy between object and scenes and claimed that object detectors emerge from training CNNs on scenes. Their “object detectors” are grandmother cell-like filters for which they provide quantitative evaluations by asking workers on Amazon Mechanical Turk (AMT) to evaluate the performance of the detector. By comparison, we show that mid-level object part structures can be extracted using population encoding representation at intermediate layers of a CNN, in a purely unsupervised manner. Moreover, we evaluate these mid-level object part structures by testing on key point annotations of object parts without human labor.

3 POPULATION ENCODING

In this section, we describe our approach for finding distributed representations in CNNs. We restrict ourselves to object-CNNs and show the parts can be discovered automatically (but our method could be applied to any type of CNN). We start by using a CNN which has been pre-trained on ImageNet. We then apply the deep network to a set of images from a specific object class, e.g., cars. At each level of the CNN we cluster the filter vector responses using K-means++ (Arthur & Vassilvitskii (2007)). This yields a dictionary of clusters at each level, which we call visual concepts. When visualized, these visual concepts correspond to parts and subparts of the object. This not only throws light on CNNs by finding internal representations for parts but also can be thought of as an unsupervised method for learning object parts.

3.1 DICTIONARY LEARNING BY CLUSTERING

For an image $\mathbf{x} \in \mathbb{R}^{W \times H \times 3}$ with spatial resolution $W \times H$, an intermediate layer, ℓ , produces a response, $\mathbf{f}_\ell \in \mathbb{R}^{W_\ell \times H_\ell \times N_\ell}$, where $W_\ell \times H_\ell$ is the spatial resolution of the intermediate layer ℓ ’s response and N_ℓ is the number of filters in that layer. We randomly sample a number of population responses, $\mathbf{p}_\ell^{i,j} \in \mathbb{R}^{N_\ell}$, over the spatial grid (i, j) of the layer’s response, \mathbf{f}_ℓ . Each $\mathbf{p}_\ell^{i,j}$ corresponds

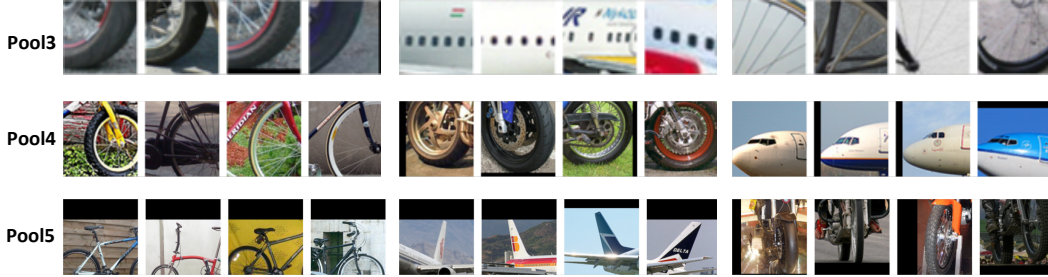


Figure 1: This figure shows example visual concepts learnt by population encoding on different object categories, for layers pool3, pool4 and pool5. Each row visualizes three concepts and four example patches are shown for each concept. These concepts are visually tight and we can identify the parent object class pretty easily by just looking at the concepts.

to a patch from the original image with the theoretical receptive field, which can be computed for any intermediate layer ℓ . We extract the population responses from all images in the training dataset.

We assume that similar image patches, with some variations in appearance in geometry, will lead to very similar filter responses. Hence we seek to identify image patches that frequently occur in car images, such as tires or license plates. Our strategy is to cluster the population responses, $\mathbf{p}_\ell^{i,j}$, using the K-means++ algorithm. At each level of the CNN, this gives a dictionary of cluster centers which we call visual concepts. To get some understanding of the visual concepts, recall that each population vector, $\mathbf{p}_\ell^{i,j}$, corresponds to a patch in the original image. Hence we can visualize the visual concepts by cropping patches from the original image (whose size is determined by the theoretical receptive field of the filters) to see which patches are assigned to each visual concept (i.e. whose population vector are closest to the cluster center). Figure 1 shows a visualization of some example mid-level visual concepts/patterns. We see that these patches correspond to meaningful mid-level parts of the object. The patches are randomly extracted from the top 100 patches assigned to the respective cluster center (i.e. those image patches whose population vector responses are closest to the cluster center). Another way to think of our approach is that we consider each object to be built from a collection of deformable templates, which vary in appearance and geometry, but where the filter clusters are to some extent invariant to these deformations. Clustering the population vectors helps identify these deformable templates.

An important issue is to determine a good value for the number of clusters K . This is partly influenced by the amount of data available because we risk over-fitting if we try to make K too large without having a sufficient amount of data. But it is also affected by the variability in appearance and geometry of the object. The greater the variability of the object the larger the number of deformable templates needed to represent it accurately, and so presumably the larger the number of visual concepts. But it is unclear how to measure the variability of the object in advance. We resort to a two-pronged strategy: (i) we choose a range of values for K (64, 128, 256, 512) to see how the results change with K , and (ii) we use a second stage to reduce the number of clusters by a merging algorithm. Our experimental evaluation suggest that performance only increases slowly with K and that merging can reduce the number of clusters by a factor of 1/3 (e.g., from 512 to 166) without significant decrease in performance. The details of the merging algorithm are in the next section.

3.2 MERGING CLUSTERS

The merging strategy starts by ranking clusters based on their goodness, where “good” clusters are those which are tight (i.e. have small intra-cluster distance) and are well-separated from other clusters (large inter-cluster distance). The “goodness” of a cluster, k , is computed using the Davies-Bouldin index,

$$DB(k) = \max_{m \neq k} \frac{\sigma_k + \sigma_m}{d(\mathbf{c}_k, \mathbf{c}_m)}, \quad (1)$$

where \mathbf{c}_k and \mathbf{c}_m denote the centers of clusters k and m respectively, and $d(\mathbf{c}_k, \mathbf{c}_m)$ is the Euclidean distance between the two cluster centers. σ_k and σ_k denote the average distance of all data points in clusters k and m to their respective cluster centers, i.e.,

$$\sigma_k = \frac{1}{n_k} \sum_{\mathbf{p} \in \mathcal{C}_k} d(\mathbf{p}, \mathbf{c}_k), \quad (2)$$

where \mathcal{C}_k is the set of data points that are assigned to cluster k , and n_k is the size of \mathcal{C}_k . A good cluster is the one which is compact, i.e. small σ_k , and is well-separated from other clusters i.e. large $d(\mathbf{c}_k, \mathbf{c}_m) \forall m \neq k$. Thus, the smaller the $DB(k)$ the better the cluster. To discourage the selection of trivial clusters with a single data point (or a small number of data points) assigned to it, we also consider the number of points assigned to a cluster,

$$CNT(k) = \frac{n_k}{\sum_m n_m} \quad (3)$$

while ranking it. The final score that we assign to each cluster is a weighted sum of the “goodness” measure and the normalized count,

$$Score(k) = \lambda CNT(k) - DB(k). \quad (4)$$

After having a rank of all initial clusters, we adopt a greedy method to merge similar clusters. The algorithm starts from the highest-ranked cluster, which we call the target cluster. Among all the data points in this target cluster, we obtain a radius, r , which encloses 95% of the points assigned to the target cluster. If there exists any other cluster lower in the rank with more than 50% of its data points whose distances to the center of the target cluster are smaller than r , we merge that cluster into the target cluster. We repeat this procedure until all the initial clusters have been iterated through.

4 VISUALIZATION AND EVALUATION OF VISUAL CONCEPTS

In this section we give examples of the visual concepts by showing the image patches that they correspond to. We are faced with two difficulties: (i) For each visual concept it is impractical to show all the image patches that it corresponds to because this varies from 400 to 4,000 depending on the visual concept. (ii) It is not practical to show all the visual concepts because there are too many. For example, if we set $K = 512$ for cars we still end up with 166 visual concepts after pruning which is far too many to show in a paper.

We address the first difficulty in two ways. Firstly, we show a limited number of the best fit image patches (e.g., the top 6) *and* a random sample from other patches that are well matched to the visual concept (e.g., a random sample of 6 patches from the top 500). Figure 1 shows that random samples from a visual concept are similar to each other, and Figure 3 shows that the best patches and the random samples are often very visually similar, for concepts learnt from the `pool3-pool5` layer. Secondly, we can visually quantify the limited variability of the image patches corresponding to a visual concept by: (a) performing edge detection using HED (Xie & Tu (2015)) and then taking the average of the edge maps of the top 500 patches, and, (b) directly taking the average of the (color) intensity of the top 500 patches. Examples of this averaging are given in Figure 2 and demonstrate that many of the visual concepts are tight.

Next we address the second issue (the large numbers of visual concepts). We have created a project website (<http://ccvl.stat.ucla.edu/projects/visual-concepts/>) which shows all the visual concepts for the car dataset along with the various patches associated with it. In addition, we show the average edge maps and the average intensity maps for each visual concept. Moreover, we prepared a high-level taxonomy of the visual concepts, which indicates the parts they correspond to. This is illustrated in Figure 4. Our taxonomy also shows that roughly 80 percent of them are tight. Note that although some tight concepts correspond to semantic parts (e.g., tires, license plates, etc.) other are best thought of as tight “patterns”. For more details, we encourage the reader to refer to the above webpage.

We now provide visual comparisons of the concepts learnt by population encoding and compare it with single filter (or grandmother cell) representation. Figure 3 provides example visualizations from `pool3-pool5`. We do not visualize the `pool1` and `pool2` clusters since their receptive fields are too small. The top two rows in each subfigure shows patches assigned to two clusters from population encoding and the bottom two rows show the patches producing the highest activations in two individual filters. Also, in each subfigure, the first six patches are the “best” patches (closest to the cluster center or with the highest activation for a single filter) and the final six patches are randomly sampled from a pool of patches assigned to the respective cluster/filter. A general trend to notice is that the patches assigned to clusters appear more compact (look at the minimal variation in the appearance patterns of the randomly sampled patches), as opposed to the patches that produce

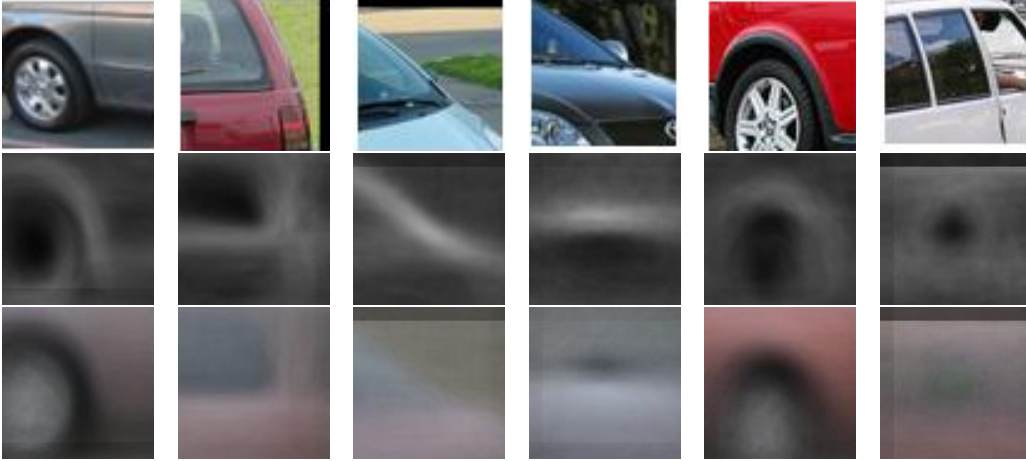


Figure 2: This figure show three example visual concepts (left three) and three single filter patches (right three). The top row contains example image patches corresponding to the visual concept and the single filter, the middle row corresponds to the average edge map obtained using 500 patches that are closest to the visual concept and the top 500 that produced the highest activations for the single filter, and, the bottom row corresponds to the mean RGB image computed using the same 500 image patches for both the visual concepts and single filters. We can see that the means of the visual concepts are sharper than the means of the single filters for both the edge map and the RGB image indicating that the visual concepts capture patterns/semantics more tightly than the single filters.

high activations for single filters. These clusters/filters achieve high AP value when tested as key point detectors by our quantitative evaluation, which will be shown Section 5 and Section 6.

In addition, the project webpage displays all the 512 filter responses for the car at `pool4`. This is analogous to the results for the visual clusters that were mentioned above. Similar to the results of visual concepts, we show the averages of the edge and intensity maps. These results show that although some filters are very tight, specifically for object tires, most of them are not tight. We estimate that roughly only 20 percent of filters are tight compared to about 80 percent of the visual concepts.

We observed that the visual clusters appeared less tight at the highest levels of the hierarchy. We conjectured that one reason for this is that the “effective receptive field” of the neurons is not the same as their theoretical receptive fields. This was discussed in Zhou et al. (2015). Hence we use a deconvolutional technique to estimate an effective receptive field for the visual concepts. We find that, for visual concepts at `pool5`, the image patches look more similar within the effective receptive field (red bounding box) than looking at the entire patch (which is very large). More precisely, we obtain the effective receptive field for both clusters and single filters by applying the DeConv operation in Zeiler & Fergus (2014), and then threshing the DeConv responses. The effective receptive field captures the region of pixels which are most responsible for the cluster or single filter response. Similar idea is in Zhou et al. (2015) using a different method. We can easily see that the effective receptive field of `pool5` is much smaller than the corresponding theoretical receptive field size.

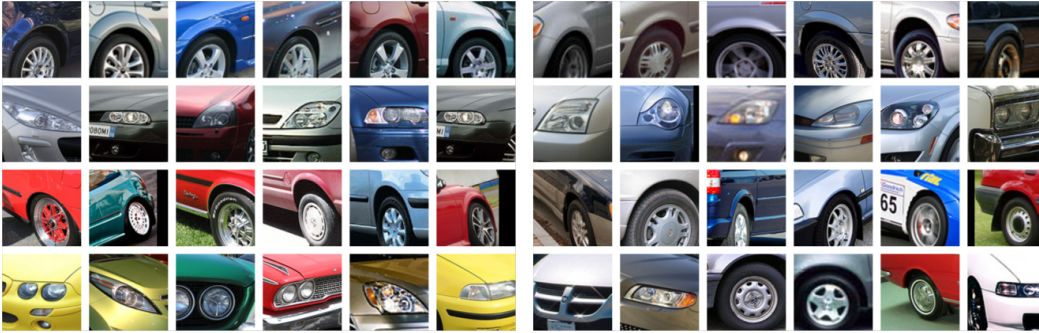
5 UNSUPERVISED PART LEARNING AND KEYPOINT DETECTION

We now validate visual concepts in an alternative manner by investigating their ability to perform unsupervised part detection. To do this we require an annotated dataset with parts of objects labelled. We choose PASCAL3D+ which annotated some objects from ImageNet and PASCAL using 3D models. The advantage of this dataset is that it includes 3D orientation and has keypoints specified on the objects. Hence it enables us to give quantitative evaluation of the visual concepts as unsupervised part detectors and to compare them to alternatives, such as supervised methods (which know the location of the keypoints) and to detection by single filters.

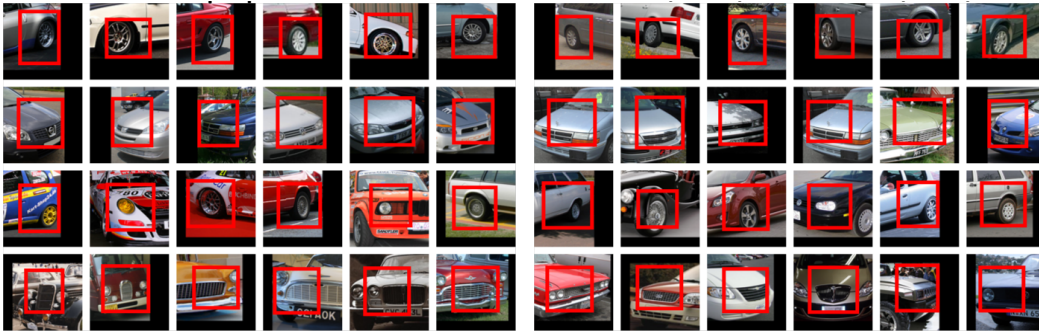
There are, however, several problems. Firstly, PASCAL3D+ only labels a limited number of key points (e.g., roughly 10) and so, many of our visual concepts do not respond strongly to them. Secondly, several key points correspond to parts that are impossible to distinguish using only local



(a) pool13 visualization



(b) pool14 visualization



(c) pool15 visualization

Figure 3: Visual comparison between the dictionaries by population encoding (visual concepts) and single filters for layers pool13, pool14 and pool15. Top two rows correspond to visual concepts and bottom two rows correspond to single filters. The patches assigned to visual clusters are more compact and visually similar than the patches that produce high activations for single filters.

image patches (e.g., front and back wheels) although this can be partly addressed by merging the key points into more general classes (e.g. wheels). Thirdly, we do not know the correspondence between the visual concepts and the key points. We now explain the evaluation strategy and we showcase the results in the next section.

Since the visual concepts correspond to object parts, we aim to test their ability to act as object part detectors. We make use of the precision/recall evaluation method and calculate the average precision (AP) of each concept for multiple detection thresholds. We test each concept against each key point and assign the one with the best AP as the detector for that key point. We use a simple nearest neighbor approach for computing the distance between cluster centers (concepts) and test patches by their feature vectors of the corresponding layer. We use this distance as the detection score for the corresponding concept. The decision to adopt the nearest neighbor strategy and not a more sophisticated detection algorithm is intended to show how good the visual concepts are for representing parts. It is highly likely that more sophisticated detection methods would lead to improved results, but we leave this for future study.

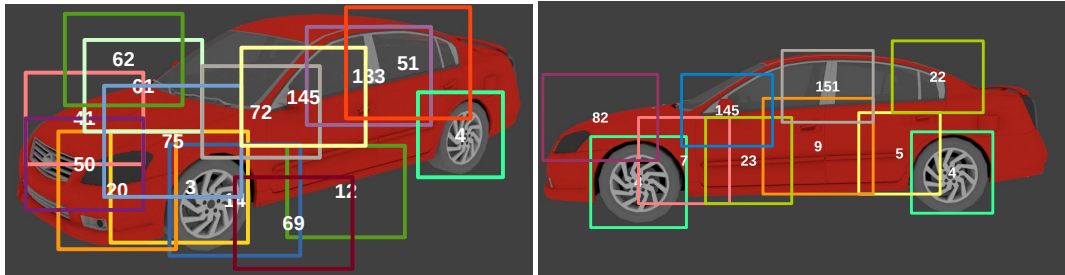


Figure 4: The figure shows two example car views rendered from a 3D model. We overlay the image with rectangles that are labeled with the visual concept number (at the center), which would be the best match at that location. We can see that the visual concepts that we have learned cover the entire car for both these example renderings. We request the reader to look at the project webpage to visualize the patches from the training set corresponding to the numbered visual concept.

If the detected patch center is sufficiently close to the key point position (within a threshold), we treat it as a true positive. In order to suppress duplicate detections, we use non-maximal suppression (NMS). From the visualizations in the previous section, the fourth layer clusters largely capture the semantic level of key point annotations in the dataset that we use (PASCAL3D+). Moreover, the theoretical receptive field of `pool4` allows for a fair spread of the patches on the objects corresponding to the density of the key point annotations in the dataset. Therefore we decide to use the `pool4` dictionaries to obtain quantitative evaluation results for key point detection, as presented in the next section.

6 EXPERIMENTAL RESULTS

Dataset: Evaluating mid-level parts requires us to use a dataset with part-level annotations. Therefore, we use annotations from PASCAL3D+ dataset (Xiang et al. (2014)) for our evaluation. This dataset augments PASCAL VOC2012 images with the images from the ImageNet dataset and annotates the objects with 2D key point positions for 12 rigid object categories (e.g. cars, aeroplanes, bicycles, etc.). In this work, we experiment on six object categories: car, aeroplane, bicycle, motorbike, bus, and train. For each category, we split the ImageNet images into two halves and use one half for learning the dictionaries and the other half for testing. Since we are trying to learn the mid-level representations of object parts, we crop the objects using the bounding box annotations and use only the cropped-out regions of the images.

The key point annotations from PASCAL3D+ are side-dependent, e.g., the left-front tires and the right-front tires are labeled with different labels. However, the receptive field of a CNN’s intermediate layer is not big enough to make such fine-grained distinctions. Therefore, we merge the key point annotations into a coarser level of granularity. The merging is done on the following occasions. a) We merge the left/right fine-scale distinctions (e.g. left/right headlight) into a single class (e.g. headlight). b) We group visually similar key points into a single class (e.g. the four wheels of a car). And, c) we discard key points that rarely occur in the images.

Baselines: We provide two baselines for comparison. i) We test for the presence of grandmother cell-like neurons and evaluate the detection performance of each single filter independently, while trying to detect the mid-level key points. We use the neuron responses as the detection scores when computing AP values. We report results for the best performing neurons. ii) To provide an upper-bound on the detection capacity of unsupervised part-learning techniques, we report results obtained by using strong supervision. We extract the population responses for patches centered at the key point annotation (and randomly sampled patches for a separate background class) and train a multi-class linear SVM to discriminate the different key point patches. Note that this baseline is very similar to the keypoint prediction method in Long et al. (2014) but with different performance measure.

Implementation Details: In the experiments, we use the Caffe toolbox (Jia et al. (2014)) and VGG-16 network Simonyan & Zisserman (2015) pre-trained on ImageNet objects as our object-CNN. We build separate dictionaries for different object classes. For each image, we crop the object using the bounding box annotation and resize the short side to 224 pixels. We normalize each population response (feature vector) using L2-normalization for both dictionary clustering and detection, since

	Car								Bicycle						Motorbike				
	1	2	3	4	5	6	7	mAP	1	2	3	4	5	mAP	1	2	3	4	mAP
single-filter	.86	.44	.30	.38	.19	.33	.13	.38	.26	.65	.32	.45	.53	.44	.25	.56	.35	.11	.31
cluster-merge	.92	.51	.27	.41	.36	.46	.18	.45	.36	.78	.39	.53	.72	.56	.33	.64	.45	.23	.41
cluster-full	.94	.51	.32	.53	.36	.48	.22	.48	.35	.80	.39	.56	.70	.56	.36	.64	.52	.37	.47
strong	.97	.65	.37	.76	.45	.57	.30	.58	.41	.80	.29	.80	.76	.61	.31	.65	.47	.31	.44

	Bus							Train						Aeroplane					
	1	2	3	4	5	6	mAP	1	2	3	4	5	mAP	1	2	3	4	5	mAP
single-filter	.45	.40	.23	.34	.80	.21	.41	.39	.32	.26	.15	.17	.26	.32	.21	.20	.14	.31	.24
cluster-merge	.42	.53	.20	.32	.87	.52	.48	.39	.35	.32	.23	.23	.30	.19	.22	.24	.12	.32	.22
cluster-full	.44	.52	.24	.32	.87	.52	.49	.42	.35	.32	.25	.23	.31	.22	.26	.24	.14	.33	.24
strong	.68	.67	.51	.58	.91	.56	.65	.66	.50	.46	.33	.33	.46	.42	.27	.34	.19	.38	.33

Table 1: This table provides the best AP values obtained while trying to detect object part key points. The rows correspond to the various detection methods. ‘single filter’ refers to the baseline method of using single filters as part detectors. ‘strong’ refers to the strong supervision baseline of training linear SVMs for each part. ‘cluster-merge’ and ‘cluster-full’ refer to two variants using population encoding. The key point number-name mapping is provided below. Cars – 1: wheel 2: wind shield 3: rear window 4: headlight 5: rear light 6: front 7: side; Bicycle – 1: head center 2: wheel 3: handle 4: pedal 5: seat; Motorbike – 1: head center 2: wheel 3: handle 4: seat; Bus – 1: front upper corner 2: front lower corner 3: rear upper corner 4: rear lower corner 5: wheel 6: front center; Train – 1: front upper corner 2: front lower corner 3: front center 4: upper side 5: lower side; Aeroplane – 1: nose 2: upper rudder 3: lower rudder 4: tail 5: elevator and wing tip.

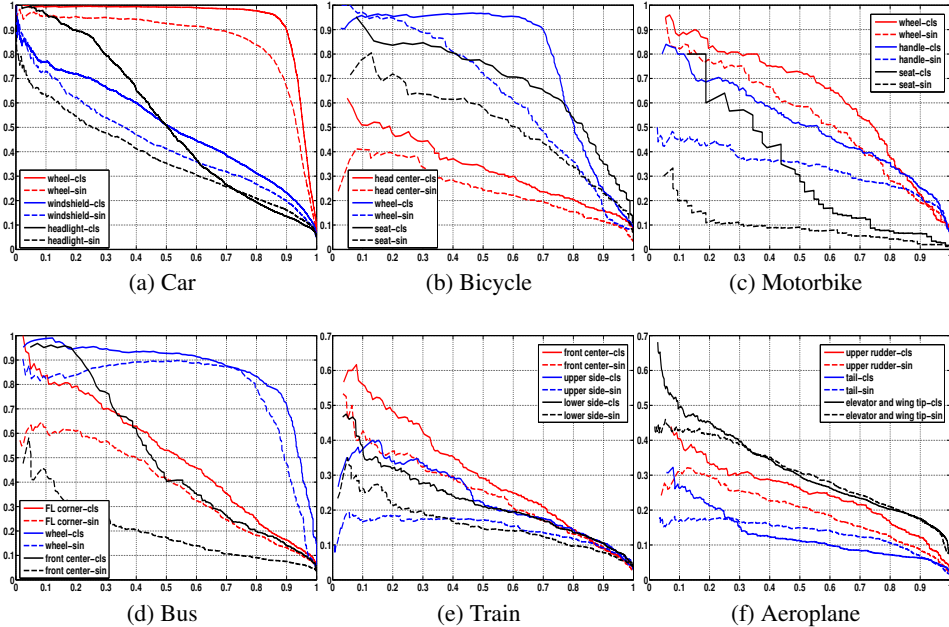


Figure 5: This figure plots the PR curves of some typical key points for six rigid object categories. Notice that population encoding performs better than single filters in all of these examples.

we find that L2-normalization gives much better results. We do not use L2-normalization for single neuron evaluation. We analyse the effect of normalization on the supplementary material.

Quantitative Results: While evaluating the dictionaries on the key point annotations, it is important to select the dictionary layer carefully. Based on our visualizations, we find that the `p0014` dictionaries best capture the level of semantic annotations in the PASCAL3D+ dataset. Therefore, we primarily report the results using the dictionaries learnt from `p0014`. In Section 6.1 we perform some diagnostic experiments where we report results from other layers.

Table 1 reports the various average precision (AP) values that we obtain. For each key point from a particular object category, we show the AP achieved by the best visual concept and the best single

neurons. By ‘single filter’ we refer to the baseline we obtain when testing single filters as detectors and ‘cluster’ refers to the use of visual concepts as detectors. ‘cluster-merge’ refers to case where we use merging to reduce the number of clusters, while ‘cluster-full’ refers to the unmerged case. Finally, ‘strong’ refers to the baseline where we use strong supervision to train linear SVMs to act as key point detectors. In Figure 5 we plot the PR curves for some key points of all the six object categories.

The general trend that we observe from the numbers in the table is that the population encoding (visual concepts) is a better way to encode the mid-level representations than single neurons. The merged clusters performs slightly worse than the full clusters, since merged clusters might not be as pure and representative as the unmerged clusters. However, merged cluster provides a much more compact representation due to their smaller dictionary size. Also we can see that the visual concepts obtained in an unsupervised way by population encoding achieve reasonably good AP values for most categories. The only exception is for Aeroplane, which is difficult due to large elevation and viewpoint variations. Observe that all methods have low AP values on aeroplanes.

6.1 DIAGNOSTIC EXPERIMENTS

In this subsection we present results from some diagnostic experiments that we conduct to enable a better understanding of the visual concepts obtained from population encoding. Currently, our diagnostic experiments are performed on cars. Experiments on other object categories is left as future work.

Effect of the number of clusters K : We investigate the effect of varying K on the performance. Our results, Table 2, show that although performance increases with K the improvement is relatively slow from $K = 64$ to $K = 512$. Note that even at $K = 64$ the performance is still better than those of the best filters (note that there are 512 filters on Pool4).

Method	1	2	3	4	5	6	7	mAP
cluster-512	.94	.51	.32	.53	.36	.48	.22	.48
cluster-256	.94	.47	.29	.54	.31	.45	.20	.45
cluster-128	.94	.46	.27	.51	.17	.47	.19	.43
cluster-64	.93	.47	.27	.29	.17	.46	.18	.40
single-filter	.86	.44	.30	.38	.19	.33	.13	.38

Table 2: The effect of K in clustering. “cluster- K ” refers to setting K as the number of cluster in kmeans. All the cluster-based methods (visual concepts) are without merging.

Effect of Intermediate Layers: In Table 1, we present the results using the dictionary elements learnt from the pool4 layer. We select the pool4 dictionaries since they correspond well to the semantic level of the annotated key points in the PASCAL3D+ dataset. In Table 3, we present the results obtained using the dictionaries learnt from pool3 and pool5 layers. Looking at the numbers from pool3, it is clear that its concept dictionaries are not able to capture semantic level of the part annotations. This is because the receptive field of pool3 is too small to capture the entire part. Surprisingly, pool5 gives similar results as pool4 layer. But on careful examination, we see that pool5 does better than pool4 when the key points correspond to “big” parts (e.g. windshield) and worse than pool4 when the key points correspond to smaller parts (e.g. wheels), the size of which is more appropriate for pool4. This shows that different layers of the CNN are capturing different levels and scales of object parts. It is interesting to see how pool5 layer dictionaries perform on other object categories, and we will do this in future work.

Viewpoint Control: While visualizing the clusters, we find that many visual clusters are viewpoint specific. For example, front-view wheels and the side-view wheels are in separate clusters. Therefore, we test the performance of the concepts by controlling for viewpoint. We divide the test set into five viewpoint bins: front, front-side, side, rear-side, and rear. Then, for each viewpoint we run the same evaluation procedure as described above. Table 4 shows the best viewpoint result for each key point. The numbers clearly shows a significant jump in detection capabilities of both single filters and our visual concepts (population encoding still outperforms single-filter detectors). These results point to a lack of viewpoint invariance in the internal representation of the CNNs.

	1	2	3	4	5	6	7	mAP
pool3	.80	.28	.13	.22	.26	.22	.16	.29
pool5	.83	.61	.27	.55	.29	.45	.29	.47

Table 3: AP values obtained using dictionaries learnt from two other layers on the Car key points. Please refer to the Table 1’s caption for key point number-name mapping.

Method	1	2	3	4	5	6	7	mAP
single-filter	.96	.65	.61	.59	.59	.38	.27	.58
cluster-merge	.99	.75	0.63	.61	.70	.49	.39	.65
cluster-full	.99	.75	0.61	0.73	.70	.54	.39	.67
strong	.99	.87	.82	.88	.84	.62	.41	.78

Table 4: AP values after controlling for view-point on cars. Please refer to Table 1’s caption for key point number-name mapping.

7 FURTHER STUDIES

7.1 TIGHT CLUSTER WITH LOW AVERAGE PRECISION

When examining the clusters with low AP value, we find that many of them are tight clusters. This means that these clusters capture visual patterns very well but they do not correspond to semantic key points. Figure 6 illustrates this face. On the left is an image of a car. The two bounding boxes are detections of two different clusters, which are visualized on the right with six patches. We can see that the two clusters capture horizontal structures and textures respectively, but they do not match the key point annotations in the dataset and thus have low AP values. This is due to the limitation of the current key point annotations. In order to fully evaluate the clusters of different layers, we should have key point annotations with high degree of coverage and at different semantic levels.

7.2 COMPOSITIONAL RELATION BETWEEN ADJACENT LAYERS

Having obtained the mid-level parts from the population encoding method, we can further discover the compositional relations between adjacent layers (e.g., `pool3` and `pool4`). If a cluster from a higher layer fires (gets detected), we can calculate which clusters in the lower layer also fire within the higher layer cluster’s receptive field. In other words, we want to compute the conditional probability of lower layer clusters firing spatially nearby given the firing of some higher layer cluster. Figure 7 shows two examples of such compositional relations between `pool4` and `pool3`. We can see that the selected top `pool3` clusters capture parts of the patterns that the `pool4` cluster represents. Note that we can apply this compositional analysis to any adjacent layer pairs.

8 CONCLUSION AND FUTURE WORK

In this paper, we hypothesized that CNNs represent information about object parts using neural populations involving the co-activity of several filters. To obtain these internal representation we used a simple clustering technique to learn dictionaries of visual concepts at different levels of the CNN. We find these internal representations for each object separately and our approach can be thought of as unsupervised learning, in the sense that the part locations and the number of parts is unknown (although the object identity is). We used visualization techniques to show that visual concepts are often tight (i.e. they respond to visually similar patches), correspond to most of the parts of the object class, and perform better than individual filters. We also perform quantitative experiments on PASCAL3D+ showing that the visual clusters perform well as (unsupervised) key point detectors, comparing favorably with individual neurons and only a bit worse than fully supervised methods. Overall, our work gives some understanding of the internal representations used by object CNNs.

In future work, we plan to apply this approach to CNNs performing other visual tasks, evaluate it on more complete datasets (PASCAL3D+ has a limited number of key points and viewpoints), and investigate the co-activation of visual concepts at different levels of the hierarchy. An interesting possibility is to build on the ideas in (Zhu et al. (2010b) and Zhu et al. (2010a)) to learn compositional models, by hierarchically grouping visual concepts which frequently co-occur, which can parse new objects automatically.

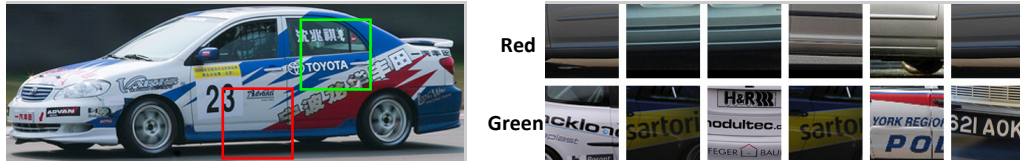


Figure 6: Two tight clusters which do not correspond to key points well.

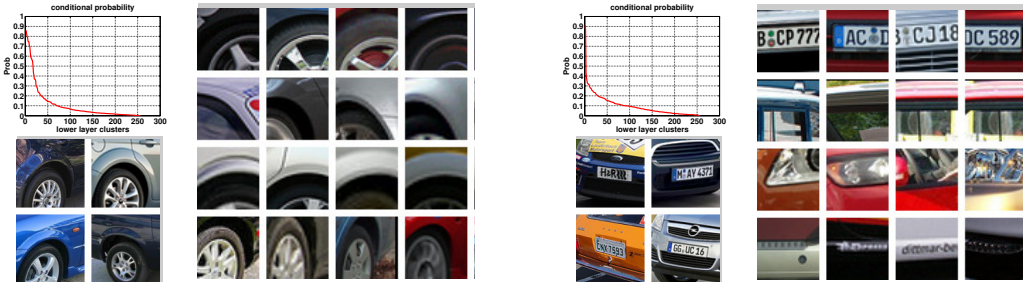


Figure 7: This figure provides two examples of the compositional relations between a `pool4` cluster and many `pool3` clusters. For each example, Top Left: conditional probability, Bottom Left: four patches from a `pool4` cluster, Right: top-4 `pool3` clusters that occurs most frequently within the considered `pool4` cluster's receptive field. Each row corresponds to a cluster and for each cluster we show four example patches.

ACKNOWLEDGMENTS

We would like to thank Junhua Mao and Roozbeh Mottaghi for fruitful discussions during the initial stages of the project. We would also like to thank Xiaochen Lian and Weichao Qiu for help with rendering cars from 3D models using the Blender software.

REFERENCES

- Agrawal, Pukit, Girshick, Ross B., and Malik, Jitendra. Analyzing the performance of multilayer neural networks for object recognition. In *ECCV*, pp. 329–344, 2014.
- Arthur, David and Vassilvitskii, Sergei. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pp. 1027–1035, 2007.
- Barlow, H. Single units and sensation: A neuron doctrine for perceptual psychology? *Perception*, 1:371–394, 1972.
- Bourdev, Lubomir D., Maji, Subhransu, Brox, Thomas, and Malik, Jitendra. Detecting people using mutually consistent poselet activations. In *ECCV*, pp. 168–181, 2010.
- Chen, Liang-Chieh, Papandreou, George, Kokkinos, Iasonas, Murphy, Kevin, and Yuille, Alan L. Semantic image segmentation with deep convolutional nets and fully connected crfs. In *ICLR*, 2015.
- Erhan, Dumitru, Bengio, Yoshua, Courville, Aaron, and Vincent, Pascal. Visualizing higher-layer features of a deep network. In *Technical Report 1341*. 2009.
- Georgopoulos, Apostolos P, Schwartz, Andrew B, and Kettner, Ronald E. Neuronal population coding of movement direction. *Science*, 233(4771):1416–1419, 1986.
- Girshick, Ross B., Donahue, Jeff, Darrell, Trevor, and Malik, Jitendra. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, pp. 580–587, 2014.
- Jia, Yangqing, Shelhamer, Evan, Donahue, Jeff, Karayev, Sergey, Long, Jonathan, Girshick, Ross, Guadarrama, Sergio, and Darrell, Trevor. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

- Juneja, Mamta, Vedaldi, Andrea, Jawahar, CV, and Zisserman, Andrew. Blocks that shout: Distinctive parts for scene classification. In *CVPR*, pp. 923–930. IEEE, 2013.
- Krizhevsky, Alex, Sutskever, Ilya, and Hinton, Geoffrey E. Imagenet classification with deep convolutional neural networks. In *NIPS*, pp. 1106–1114, 2012.
- Li, Yao, Liu, Lingqiao, Shen, Chunhua, and van den Hengel, Anton. Mid-level deep pattern mining. In *CVPR*, pp. 971–980, 2015.
- Long, Jonathan, Shelhamer, Evan, and Darrell, Trevor. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015.
- Long, Jonathan L, Zhang, Ning, and Darrell, Trevor. Do convnets learn correspondence? In *Advances in Neural Information Processing Systems*, pp. 1601–1609, 2014.
- Mahendran, Aravindh and Vedaldi, Andrea. Understanding deep image representations by inverting them. In *CVPR*, pp. 5188–5196, 2015.
- Nguyen, Anh Mai, Yosinski, Jason, and Clune, Jeff. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *CVPR*, pp. 427–436, 2015.
- Simon, Marcel and Rodner, Erik. Neural activation constellations: Unsupervised part model discovery with convolutional networks. *CoRR*, abs/1504.08289, 2015.
- Simonyan, Karen and Zisserman, Andrew. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.
- Simonyan, Karen, Vedaldi, Andrea, and Zisserman, Andrew. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *ICLR Workshop*, 2014.
- Singh, Saurabh, Gupta, Abhinav, and Efros, Alexei. Unsupervised discovery of mid-level discriminative patches. In *ECCV*, pp. 73–86. Springer, 2012.
- Szegedy, Christian, Zaremba, Wojciech, Sutskever, Ilya, Bruna, Joan, Erhan, Dumitru, Goodfellow, Ian J., and Fergus, Rob. Intriguing properties of neural networks. *CoRR*, abs/1312.6199, 2013.
- Xiang, Yu, Mottaghi, Roozbeh, and Savarese, Silvio. Beyond pascal: A benchmark for 3d object detection in the wild. In *WACV*, pp. 75–82, 2014.
- Xiao, Tianjun, Xu, Yichong, Yang, Kuiyuan, Zhang, Jiaxing, Peng, Yuxin, and Zhang, Zheng. The application of two-level attention models in deep convolutional neural network for fine-grained image classification. In *CVPR*, pp. 842–850, 2015.
- Xie, Saining and Tu, Zhuowen. Holistically-nested edge detection. In *Proceedings of IEEE International Conference on Computer Vision*, 2015.
- Yosinski, Jason, Clune, Jeff, Nguyen, Anh Mai, Fuchs, Thomas, and Lipson, Hod. Understanding neural networks through deep visualization. In *ICML Deep Learning Workshop*, 2015.
- Zeiler, Matthew D and Fergus, Rob. Visualizing and understanding convolutional networks. In *ECCV*, pp. 818–833. 2014.
- Zeiler, Matthew D, Taylor, Graham W, and Fergus, Rob. Adaptive deconvolutional networks for mid and high level feature learning. In *ICCV*, pp. 2018–2025. IEEE, 2011.
- Zhou, Bolei, Khosla, Aditya, Lapedriza, Àgata, Oliva, Aude, and Torralba, Antonio. Object detectors emerge in deep scene cnns. In *ICLR*, 2015.
- Zhu, Long, Chen, Yuanhao, Torralba, Antonio, Freeman, William, and Yuille, Alan. Part and appearance sharing: Recursive compositional models for multi-view. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pp. 1919–1926. IEEE, 2010a.
- Zhu, Long, Chen, Yuanhao, and Yuille, Alan. Learning a hierarchical deformable template for rapid deformable object parsing. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(6):1029–1043, 2010b.

SUPPLEMENTARY MATERIAL

A. L2-NORMALIZATION

In this subsection, we make a comparison between using L2-normalization as a preprocessing step and without L2-normalization. From Table 5, we can see that L2-normalization is critical for clusters from population encoding strategy. We then hypothesize that it is the population response direction that captures visual information, instead of the population response magnitude. For the single neuron case, we have also tried similar L2-normalization across filters for each spatial position, and used the normalized response as detection score. But we find that L2-normalization in single neuron case makes little difference (slightly worse) in terms of the AP values.

Method	1	2	3	4	5	6	7	mAP
cluster-full-NL2	.92	.33	.17	.34	.12	.39	.16	.35
cluster-full-L2	.94	.51	.32	.53	.36	.48	.22	.48
single-filter-NL2	.86	.44	.30	.38	.19	.33	.13	.38
single-filter-L2	.80	.42	.24	.42	.19	.31	.18	.37

Table 5: The effect of L2-normalization for cluster and single filter. Results are obtained using imagenet cars. ‘NL2’ refers to not using L2-normalization, and ‘L2’ refers to using L2-normalization.

B. RESULTS ON PASCAL VOC

In this section, we provide the evaluation results of both the population encoding and single filter on cars and aeroplanes from Pascal VOC dataset. Since we only consider non-occluded bounding box objects, there is limited number of non-occluded objects in Pascal for cars and aeroplanes. So we use all the Pascal data for test, while using the dictionaries trained on imagenet for the corresponding category. We use key point annotations from Bourdev et al. (2010) since they provide much richer key points. The AP values are shown in Table 6 and PR curves are shown in Figure 8. We can see that methods using population encoding are better than single neuron encoding, while the strong supervision method always achieves the best AP value.

	Car										Aeroplane									
	1	2	3	4	5	6	7	8	mAP		1	2	3	4	5	6	7	8	9	mAP
single-neuron	.68	.42	.33	.32	.18	.21	.10	.31	.32		.31	.30	.32	.26	.25	.17	.35	.26	.28	.28
cluster-merge	.79	.50	.44	.32	.30	.27	.16	.38	.40		.28	.52	.37	.33	.38	.21	.37	.33	.36	.35
cluster-full	.79	.48	.38	.46	.35	.32	.18	.44	.43		.30	.55	.41	.32	.32	.26	.41	.30	.36	.36

Table 6: Evaluation results on Pascal cars and aeroplanes. The key point number-name mapping is provided below. Cars – 1: wheel 2: wind shield 3: rear window 4: headlight 5: back trunk 6: front 7: side 8: mirror; Aeroplanes – 1: nose tip 2: upper nose 3: lower nose 4: upper rudder 5: lower rudder 6: wing base 7: engine front 8: engine back 9: elevator and wing tip.

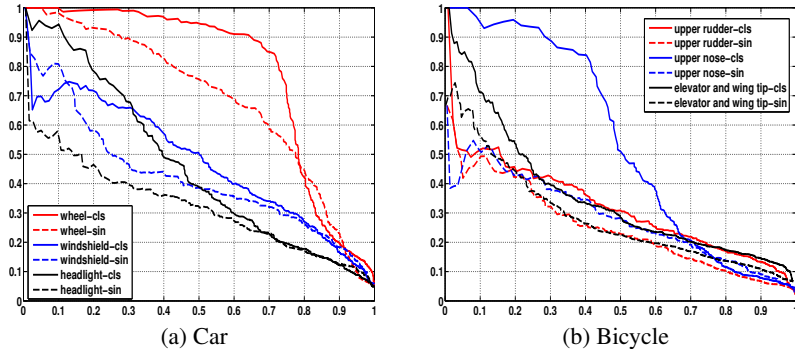


Figure 8: This figure plots the PR curves of some typical key points for Pascal cars and aeroplanes.